REMARKS

Claims 1-9 and 16-21 are pending. Claims 1, 5, 7, and 8 are amended. Claims 10-15 are canceled. Claims 16-21 are added. Support for the amendments and new claims 16-21 may be found on at least page 8, line 25, to page 16, line 2, of the originally filed specification. Reconsideration of the claims is respectfully requested in view of the following remarks.

I. Title of the Invention

The Title of the Invention is amended based on the Examiner's suggestion.

II. 35 U.S.C. § 102, Alleged Anticipation of Claims 1-5, 7, and 8

The Office rejects claims 1-5, 7, and 8 under 35 U.S.C. § 102(b) as allegedly being anticipated by *Sollars* (U.S. Patent No. 5,900,025). This rejection is respectfully traversed.

Sollars teaches a processor having a hierarchical control register file. *Sollars* teaches a primary control register file that comprises a plurality of control registers that are organized into control register sets. See *Sollars*, col. 5, lines 19-43. *Sollars* teaches that control register sets are dynamically allocated to contexts and threads during operation. See *Sollars*, col. 5, lines 43-46. *Sollars* states:

> Except for the initial allocations at system start up time, the control register sets at the various levels are dynamically allocated to the contexts and the threads during operation. Besides modifications as a result of instruction execution, i.e., status update and the like, the control registers are directly accessible and modifiable using instructions from the standard instruction sets of exemplary processor 10. A context and thread based privilege structure is employed to control the direct accesses and modifications.

Sollars, col. 5, lines 43-53. An auxiliary control register file augments the primary control register file. See *Sollars*, col. 5, lines 55-66.

Sollars also states:

> Under the presently preferred embodiment, a thread is initially conferred a standard thread privilege, which allows the thread to access and modify its own set of control

> registers. On an as needed basis, one of the threads of each context can be temporarily conferred a context privilege, which allows the context privileged thread to also access and modify its context's context level control register set as well as its peer threads' control register sets. Similarly, on an as need basis, one of the contexts (more specifically, one of the context privileged threads) can be temporarily conferred a system privilege, which allows the system privileged context to also access and modify the system level control register set as well as any one of the other "lower" level control register sets.

Sollars, col. 3, lines 45-58. See also *Sollars*, col. 15, lines 60-66. Thus, *Sollars* teaches a hierarchical control register structure where threads have associated thread privileges. A thread may access certain context levels within the control register structure based on the thread's privilege.

Sollars also teaches a primary and secondary operand register file. With respect to the primary operand register file, *Sollars* states:

> Primary operand register file 22a includes a number of registers for performing the conventional function of storing instruction operands in a new and innovative manner. Preferably, primary operand register file 22a is a scalable uni/multidimensional as well as virtually/physically addressable register file, used for storing integer as well as floating point operands, as disclosed in copending U.S. patent application, application Ser. No. 08/401,411, having common inventorship with the present invention, which is hereby fully incorporated by reference.

Sollars, col. 5, lines 21-30. However, *Sollars* does not teach or suggest that each register file of the plurality of register files at least corresponds to at least one thread of the plurality of threads, as recited in claim 1, for example.

The Office Action alleges that the primary and secondary control register files of *Sollars* are equivalent to the plurality of status and control registers and that the primary and secondary operand register files of *Sollars* correspond to the plurality of register files in claim 1. Applicant respectfully disagrees. In *Sollars*, the primary and secondary control register files comprise sets of control registers at various levels that are allocated to contexts and threads during operation. However, *Sollars* does not teach that each

control register corresponds to a thread. Rather, *Sollars* teaches that sets of control registers correspond to a privilege level.

Furthermore, *Sollars* does not teach that each operand register file corresponds to a thread. Rather, *Sollars* teaches that the primary operand register file includes a number of registers for performing the function of storing instruction operands. More specifically, *Sollars* teaches that the primary operand register file is a scalable uni/multidimensional as wells as virtually/physically addressable register file, used for storing integer as well as floating point operands. However, *Sollars* does not teach that each register file corresponds to a thread.

Still further, *Sollars* does not teach or suggest a plurality of control bit sets that are configured to allow a thread associated with an associated status and control register to utilize other register files associated with other threads. The Office Action alleges that *Sollars* teaches this function at col. 15, lines 60-66, which states:

> On an as needed basis, one of the threads of each context is temporarily conferred a context privilege, which further allows the context privileged thread to access and modify the thread's context level control register set 104 as well as the peer threads' control register sets 106.

The above portion teaches that threads may be allowed to access and modify a peer thread's control register sets. However, the Office Action alleges that the primary and secondary **operand** register files are equivalent to the plurality of register files of claim 1. Thus, the Office Action has not shown that *Sollars* teaches of fairly suggests a plurality of control bit sets that are configured to allow a thread to utilize other [operand] register files associated with other threads. In fact, *Sollars* does not even teach that each operand register file corresponds to a thread; therefore, *Sollars* cannot teach or suggest the feature of allowing one thread to utilize the register file of another thread, as recited in claim 1.

The applied reference fails to teach or fairly suggest each and every claim limitation; therefore, *Sollars* does not anticipate claim 1. Independent claim 7, as well as new claims 10 and 13, recites limitations addressed above with respect to claim 1 and is allowable for similar reasons. Since claims 2-5 and 8 depend from claims 1 and 7, the same distinctions between *Sollars* and claims 1 and 7 apply for these claims. Additionally, claims 2-5 and 8 recite further combinations of features not suggested by the reference.

Therefore, Applicant respectfully requests withdrawal of the rejection of claims 1-5, 7, and 8 under 35 U.S.C. § 102(b).

III. 35 U.S.C. § 103(a), Alleged Obviousness of Claims 6 and 9

The Office rejects claims 6 and 9 as allegedly being unpatentable over *Sollars*. This rejection is respectfully traversed.

With respect to claims 6 and 9, the Office Action alleges that Sollars teaches one bit of a doublet that is configured to correspond to a read function and one bit of the doublet being configured to correspond to a write function, where the bits of the doublet enable or disable reading to or writing from a register file associated with a different thread, at col. 10, lines 51-67. The cited portion of Sollars states:

> As shown in FIG. 9a, TCTL control register 112c is used to store a number of control values for controlling the execution of the thread. For the illustrated embodiment, TCTL control register 112c comprises a number of flag values (AFE, MFE, SFE, and LFE) denoting whether update of the HWFLAG control register 112e is to be enabled/disabled for various instructions (i.e. Add, Multiply and Accumulate, Shift and Logic). Additionally, TCTL control register 112c comprises an sl value denoting whether the destination cache line of a store operation is to be locked, an sc value denoting whether the store data of a store operation is cacheable, and an sb value denoting whether store operations are to be forced to complete in program order. TCTL control register 112c further comprises an ll value denoting whether the destination cache line of a load operation is to be locked, an lc value denoting whether the load data of a load operation is cacheable, and an lb value denoting whether load operations are to be forced to complete in program order. Lastly, TCTL control register 112c comprises a clk_div value for halting execution of the particular thread.

Sollars, col. 10, lines 45-64. Applicant respectfully disagrees. As stated above, Sollars does not teach or suggest that each operand register file corresponds to a thread. Sollars also fails to teach or suggest a plurality of control bit sets that are configured to allow a thread associated with an associated status and control register to utilize other register files associated with other threads. Sollars does appear to teach a TCTL control register that includes flags for enabling various functions, such as locking a cache line, setting

whether a store or load operation is cacheable, or setting whether a store or load operation is forced to complete in program order. However, the Office Action proffers no analysis as to why this is somehow equivalent to separately enabling a thread to read from a register file associated with a different thread or separately enabling a thread to write to a register file associated with a different thread.

Applicant asserts that Sollars simply fails to teach or suggest such a feature. Given a lack of any suggestion in the cited prior art to provide a bit doublet in a status and control register to separately enable a thread to read from or write to a register file associated with a different thread, a person of ordinary skill in the art would not have found it obvious to modify the prior art to include this feature. The mere fact that a prior art reference can be readily modified does not make the modification obvious unless the prior art suggested the desirability of the modification. *In re Laskowski*, 871 F.2d 115, 10 U.S.P.Q.2d 1397 (Fed. Cir. 1989) and also *see In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992) and *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1993). The Office may not merely state that the modification would have been obvious to one of ordinary skill in the art without pointing out in the prior art a suggestion of the desirability of the proposed modification.

The Office Action has not shown that the prior art teaches or suggests a bit doublet in a status and control register for separately enabling a thread to write to a register file associated with a different thread. Therefore, the Office Action does not establish a *prima facie* case of obviousness. If the Patent Office does not produce a *prima facie* case of unpatentability, then without more the applicant is entitled to grant of a patent. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Grabiak*, 769 F.2d 729, 733, 226 U.S.P.Q. 870, 873 (Fed. Cir. 1985).

Therefore, Applicant respectfully requests withdrawal of the rejection of claims 6 and 9 under 35 U.S.C. § 103(a).

IV. New Claims 16-21

New claim 16 recites a method for utilizing a plurality of register files in a multithread system, the method comprising receiving an instruction for a first thread having an operations code field, a write field, and one or more read fields, wherein the operations code field defines a desired operation for the instruction, wherein the write field defines an address location to which a result of the operation is to be stored, and wherein the at least one read field defines an address location from which data is to be read for the operation; decoding the instruction; setting a first status and control register associated with the first thread and a second status and control register associated with a second thread based on the decoding of the instruction, wherein a first register file is associated with the first thread and a second register file is associated with the second thread; determining whether the first thread is permitted to utilize the second register file associated with the second thread based on at least one of the first status and control register or the second status and control register; and if the first thread is permitted to utilize the second register file, performing the operation utilizing the second register file by writing to or reading from the second register file associated with the second thread. The cited prior art does not teach or suggest decoding an instruction and setting a first status control register and a second status control register based on the decoding and determining whether a first thread is permitted to utilize a register file associated with a second thread based on at least one of the first status control register and the second status and control register. The cited prior art also fails to teach or suggest performing an operation of an instruction utilizing a register file associated with a different thread by writing to or reading from the other register file, as enabled by one or more status and control registers.

New claim 19 recites an apparatus for utilizing a plurality of register files in a multithread system, the apparatus comprising a decoder that receives and decodes an instruction for a first thread having an operations code field, a write field, and one or more read fields, wherein the operations code field defines a desired operation for the instruction, wherein the write field defines an address location to which a result of the operation is to be stored, and wherein the at least one read field defines an address location from which data is to be read for the operation; a first status and control register

associated with the first thread; a first register file associated with the first thread; a second status and control register associated with a second thread; a second register file associated with the second thread, wherein the decoder sets the first status and control register and the second status and control register based on the decoding of the instruction, wherein a first register file is associated with the first thread and a second register file is associated with the second thread; an address control that enables or disables access to the first register file and the second register file based on the first status and control register and the second status and control register, wherein the address control enables the first thread to access the second register file if the first thread is permitted to utilize the second register file; and one or more execution units that perform the operation utilizing the second register file by writing to or reading from the second register file associated with the second thread based on the address control. The cited prior art does not teach or suggest an address control that enables a first thread to utilize a register file associated with a second thread based on at least one of a first status control register and a second status and control register. The cited prior art also fails to teach or suggest one or more execution units that perform an operation of an instruction utilizing a register file associated with a different thread by writing to or reading from the other register file, as enabled by one or more status and control registers.

Since claims 17, 18, 20, and 21 depend from claims 16 and 19, these claims are allowable at least by virtue of their dependency on allowable base claims.

V. Conclusion

It is respectfully urged that the subject application is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: August 4, 2006

Stephen R. Tkacs
Reg. No. 46,430
AGENT FOR APPLICANT

Stephen J. Walder, Jr.
Reg. No. 41,534
WALDER INTELLECTUAL PROPERTY LAW, P.C.
P.O. Box 832745
Richardson, TX 75083
(214) 722-6419
ATTORNEY FOR APPLICANT